

Name: Purnendu Shekhar Achary

Branch: CSE

Year: 2<sup>nd</sup>

College: GITA Autonomous College

Course: Data Science


Mail: [purnenduachary95@gmail.com](mailto:purnenduachary95@gmail.com)

GitHub: <https://github.com/purnendux95>

---

I came across this opportunity to be a part of this course. I have completed coursework in Data Science with 2 Major Projects in hand, which has prepared me for upcoming opportunities. I am enthusiastic about the opportunity to gain hands-on experience and contribute to the community.

# MAJOR PROJECT 1



MajorProject1.ipynb ☆


File Edit View Insert Runtime Tools Help Last edited on January 11

Comment Share Settings Profile


+ Code + Text

Connect Editing

```
[ ] #MAJOR PROJECT 1
    #Name:PURNENDU SHEKHAR ACHARY
    #REGRESSION - LINEAR REGRESSION
    #Univariate/Single - 1 column as input , 1 column as output
    #Dataset - '/content/Social_Network_Ads.csv'
    #User ID , Salary in rupees
```



```
import pandas as pd
df = pd.read_csv('/content/Social_Network_income.csv')
df
```



	User ID	Salary
0	15598044	19000
1	15600575	25000
2	15603246	33000
3	15624510	43000
4	15668575	57000
5	15694829	58000
6	15727311	65000
7	15728773	76000
8	15804002	84000
9	15810944	100000



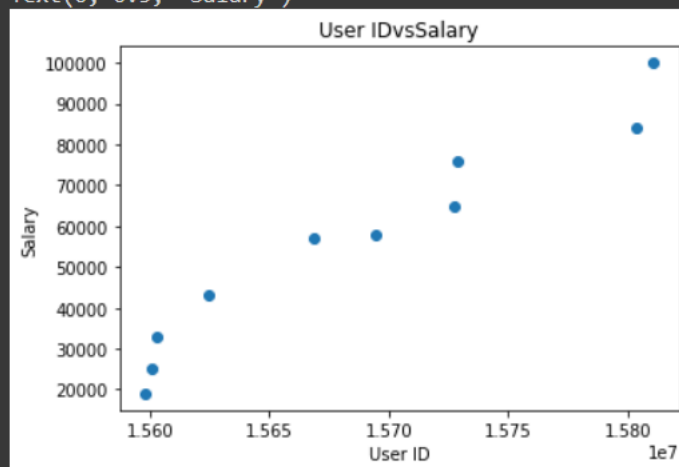
+ Code + Text

Connect

Editing

```
[ ] #3.DATA VISUALISATION - CREATION of GRAPHS
import matplotlib.pyplot as plt
#plt.scatter(x-axis,y-axis)
plt.scatter(df['User ID'],df['Salary'])
plt.title('User ID' 'vs' 'Salary')
plt.xlabel('User ID')
plt.ylabel('Salary')
```

Text(0, 0.5, 'Salary')



```
#INPUT(x) - User ID
#OUTPUT - Salary
#4.DIVIDE THE DATA INTO INPUT and OUTPUT
#INPUT(x) is always 2 dimensional,OUTPUT(y) is always 1 dimensional array
#df.iloc[row slicing,column slicing]
#In df.iloc ,if the column's place has a ':',then array is 2 dimensional
x = df.iloc[0:10,0:1].values
x
#.values converts the dataframe into an array
```



+ Code + Text

Connect ▾

Editing



```
[ ] array([[15598044],
          [15600575],
          [15603246],
          [15624510],
          [15668575],
          [15694829],
          [15727311],
          [15728773],
          [15804002],
          [15810944]])
```

```
[ ] #In df.iloc ,if the column's place does not has a ':',then array is 1 dimensiona
y = df.iloc[0:10,1].values #df.iloc[:,1].values
#If I want to select all rows or all cols , I can write only :
y
```

```
array([ 19000, 25000, 33000, 43000, 57000, 58000, 65000, 76000,
        84000, 100000])
```

```
[ ] #5.TRAIN and TEST VARIABLES
#Here we are not performing step no 5 , due to limited data
```

```
[ ] #6 Normalization or scaling is only done for multivariate datasets
#Here for univariate data, step no 6 is not required
```

```
[ ] #7 Run classifiers, REGRESSOR or cluster(applying a suitable algorithm)
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

+ Code + Text

Connect Editing ^

```
[ ] #8 fit the model(mapping or plotting the inputs whith the output with the library)
model.fit(x,y)
```

```
LinearRegression()
```

```
[ ] #Predict the output
y_pred = model.predict(x) #using the input values, we predict the output
y_pred
```

```
array([28249.33485198, 29047.14728619, 29889.09000124, 36591.84922729,
       50481.85511967, 58757.54364115, 68996.39922714, 69457.24544556,
       93170.65185151, 95358.88334828])
```

```
[ ] y #Actual output values
```

```
array([ 19000, 25000, 33000, 43000, 57000, 58000, 65000, 76000,
       84000, 100000])
```

```
[ ] #CONCLUSION
#when we compare we come to know there is huge difference
#This huge difference doesn't mean that our model has predicted wrong
#It only mean, that our model has not linear/less linear
#linearity of a model depends on the nature of the data as well as the size of the data
```

```
[ ] #INDIVISUAL PREDICTION
#I wanna know the Salary of user id 15603246
model.predict([[15603246]])
```

```
array([29889.09000124])
```

+ Code + Text

Connect  |  Editing 

```
[ ] #cross verification
    #y = mx+c i.e the eq of a straight line
    #m - slope
    #c - constant or y intercept
    #y - tangent/dependent variable
    #x - independent variable

    #From excel m = 0.31521629, C = -6654033.337384926
```

```
[ ] m = model.coef_
    m

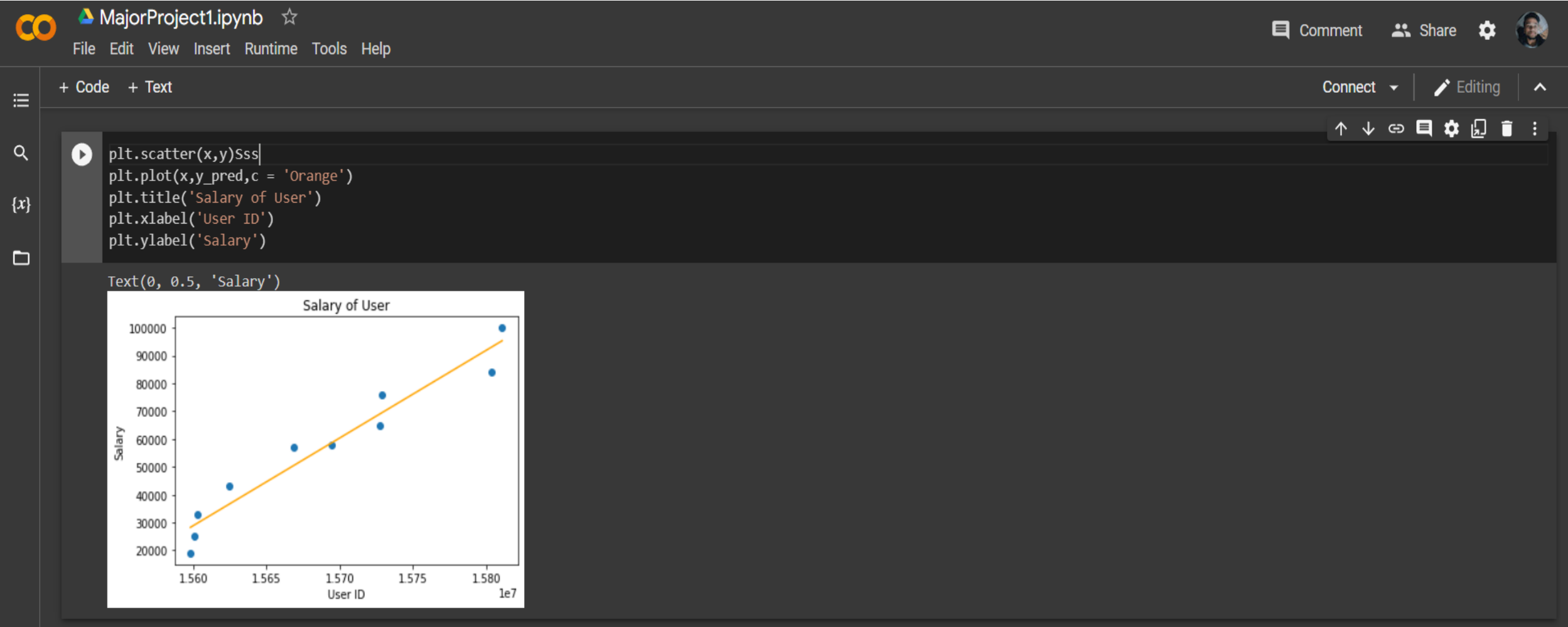
array([0.31521629])
```

```
[ ] C= model.intercept_
    C

-4888508.252115728
```

```
[ ] #y=mx+c
    m*15603246 + C

array([29889.09000124])
```



END OF MAJOR PROJECT 1

# MAJOR PROJECT 2

co

MajorProject2.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at January 12

Comment Share Settings Profile

+ Code + Text

Connect Editing

▶

#MAJOR PROJECT 2  
#NAME:PURNENDU SHEKHAR ACHARY  
#EXPLORATORY DATA ANALYSIS  
#BY USING DATASET OF BEST NETFLIX SHOWS OF ALL TIME  
#dataset = ('/content/best\_shows\_netflix.csv')

[ ]

import pandas as pd  
df = pd.read\_csv('/content/best\_shows\_netflix.csv')  
df

	title	release_year	score	number_of_votes	duration	number_of_seasons	main_genre	main_production
0	Breaking Bad	2008	9.5	1727694	48	5	drama	US
1	Avatar: The Last Airbender	2005	9.3	297336	24	3	scifi	US
2	Our Planet	2019	9.3	41386	50	1	documentary	GB
3	Kota Factory	2019	9.3	66985	42	2	drama	IN
4	The Last Dance	2020	9.1	108321	50	1	documentary	US
...	...	...	...	...	...	...	...	...
241	Evil Genius	2018	7.5	27516	48	1	crime	US
242	13 Reasons Why	2017	7.5	282373	58	4	drama	US
243	Lupin	2021	7.5	100575	46	3	crime	FR
244	All of Us Are Dead	2022	7.5	41393	61	1	action	KR
245	I Am Not Okay with This	2020	7.5	56459	21	1	comedy	US

246 rows × 8 columns





+ Code + Text

Connect ▾

Editing



```
[ ] df.shape #246 rows and 8 columns
```

```
(246, 8)
```

```
[ ] df.size #Total number of elements in the dataframe
```

```
1968
```

```
df.info() #Complete information about the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246 entries, 0 to 245
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   title                  246 non-null   object
1   release_year           246 non-null   int64
2   score                  246 non-null   float64
3   number_of_votes        246 non-null   int64
4   duration               246 non-null   int64
5   number_of_seasons      246 non-null   int64
6   main_genre             246 non-null   object
7   main_production        246 non-null   object
dtypes: float64(1), int64(4), object(3)
memory usage: 15.5+ KB
```



+ Code + Text

Connect ▾

Editing



```
[ ] #1- THE TOP FIVE SHOWS OF NETFLIX ARE:  
df.head()
```

	title	release_year	score	number_of_votes	duration	number_of_seasons	main_genre	main_production
0	Breaking Bad	2008	9.5	1727694	48	5	drama	US
1	Avatar: The Last Airbender	2005	9.3	297336	24	3	scifi	US
2	Our Planet	2019	9.3	41386	50	1	documentary	GB
3	Kota Factory	2019	9.3	66985	42	2	drama	IN
4	The Last Dance	2020	9.1	108321	50	1	documentary	US

```
▶ #2- NUMBER OF MISSING COLUMNS OR NULL VALUES OF THE SHOWS  
df.isnull().sum()
```

```
☞ title      0  
   release_year  0  
   score      0  
   number_of_votes  0  
   duration    0  
   number_of_seasons  0  
   main_genre   0  
   main_production  0  
   dtype: int64
```



+ Code + Text

Connect ▾

Editing

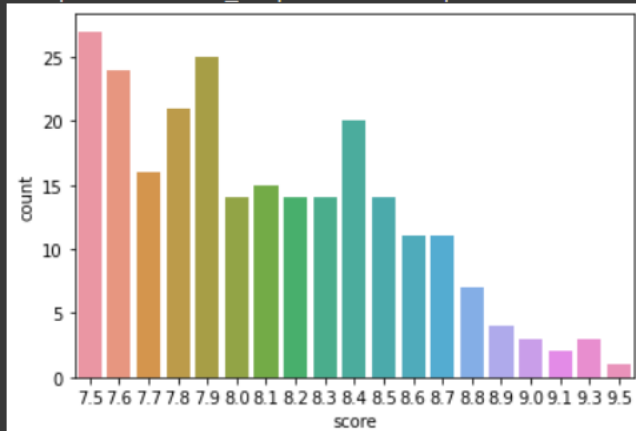


```
#3- EXACT COUNT OF UNIQUE ELEMENT OF THE SHOWS
df.nunique()
```

```
title                246
release_year         30
score                19
number_of_votes      244
duration             50
number_of_seasons    16
main_genre           12
main_production       19
dtype: int64
```

```
[ ] #VISUALISATION
#4- SHOW THE SCORE/RATINGS OF EACH SHOWS IN VISUALISATION IN INCREASING MANNER
import seaborn as sns
sns.countplot(x = 'score',data = df)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fb25c48bcd0>



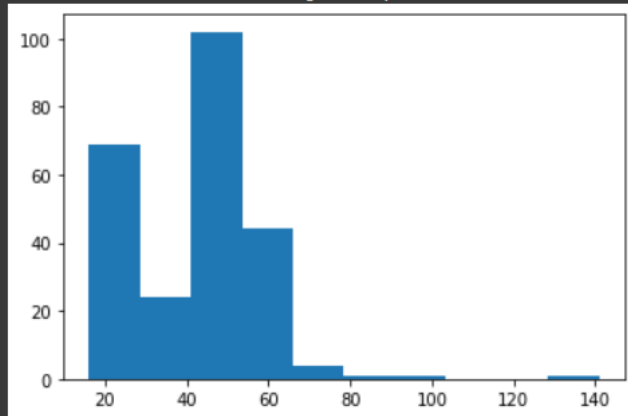


+ Code + Text

Connect | Editing

```
#5 CREATE A HISTOGRAM OF 'DURATION' COLUMN
import matplotlib.pyplot as plt
plt.hist(df['duration'])
```

```
(array([ 69.,  24., 102.,  44.,   4.,   1.,   1.,   0.,   0.,   1.]),
 array([ 16.,  28.5,  41.,  53.5,  66.,  78.5,  91., 103.5, 116.,
        128.5, 141. ]),
 <a list of 10 Patch objects>)
```



```
[ ] #6 CREATE A SCATTERPLOT OF 'DURATION' vs 'SCORE'
sns.scatterplot(x='duration', y='score', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb25becd160>
```





+ Code + Text

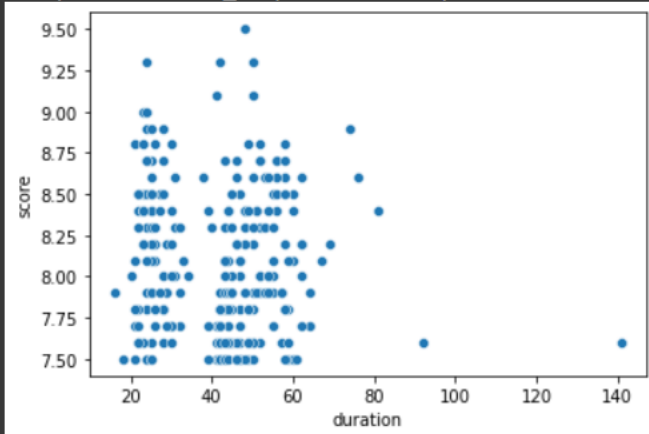
Connect ▾

Editing



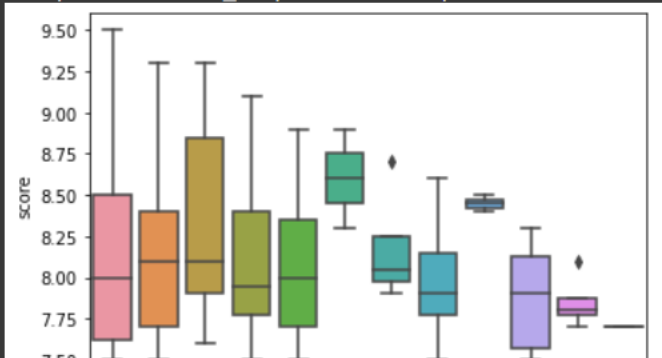
```
[ ] #6 CREATE A SCATTERPLOT OF 'DURATION' vs 'SCORE'  
sns.scatterplot(x='duration', y='score', data=df)
```

&lt;matplotlib.axes.\_subplots.AxesSubplot at 0x7fb25becd160&gt;



```
[ ] #7 CREATE A BOXPLOT OF 'SCORE' GROUPED BY 'MAIN_GENRE'  
sns.boxplot(x='main_genre', y='score', data=df)
```

&lt;matplotlib.axes.\_subplots.AxesSubplot at 0x7fb25be3ba30&gt;





+ Code + Text

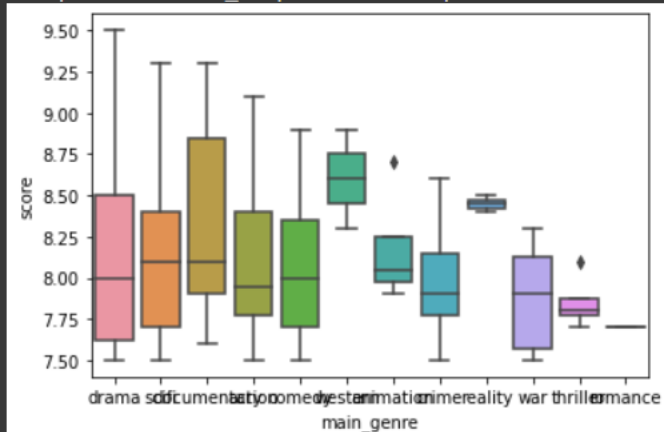
Connect ▾

Editing



```
[ ] #7 CREATE A BOXPLOT OF 'SCORE' GROUPED BY 'MAIN_GENRE'  
sns.boxplot(x='main_genre', y='score', data=df)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fb25be3ba30>



```
[ ] #8- THE NUMBER OF SHOWS WITH THEIR NUMBER OF VOTES GIVEN BY THE AUDIENCE  
df.number_of_votes.value_counts()
```

```
17992      2  
18575      2  
1727694     1  
45346      1  
22045      1  
..  
61114      1  
16970      1  
72341      1  
34362      1  
56459      1
```

```
Name: number_of_votes, Length: 244, dtype: int64
```



+ Code + Text

Connect | Editing

[ ] #9- SHOW THE UNIQUE NAMES OF THE SHOWS

df.title.unique()

```
array(['Breaking Bad', 'Avatar: The Last Airbender', 'Our Planet',  
      'Kota Factory', 'The Last Dance', 'Arcane', 'Attack on Titan',  
      'Hunter x Hunter', 'DEATH NOTE', 'Seinfeld', 'Cowboy Bebop',  
      'Heartstopper', 'When They See Us', 'Monty Python's Flying Circus',  
      'BoJack Horseman', 'Chappelle's Show', 'Better Call Saul',  
      'Narcos', 'One Piece', 'Peaky Blinders', 'Anne with an E', 'Dark',  
      'House of Cards', 'Demon Slayer: Kimetsu no Yaiba',  
      'Stranger Things', 'One-Punch Man', 'The Crown',  
      'Arrested Development', 'Friday Night Lights', 'Downton Abbey',  
      'Code Geass: Lelouch of the Rebellion', 'Trailer Park Boys',  
      'Mindhunter', 'The Haunting of Hill House', 'The Queen's Gambit',  
      'Cobra Kai', 'Wentworth', 'It's Okay to Not Be Okay',  
      'Making a Murderer', 'Shameless', 'Sacred Games',  
      'Formula 1: Drive to Survive', 'Queer Eye', 'Community',  
      'The Last Kingdom', 'Schitt's Creek', 'Neon Genesis Evangelion',  
      'Call the Midwife', 'The IT Crowd', 'ERASED', 'Supernatural',  
      'Ozark', 'Delhi Crime', 'After Life', 'Hilda', 'Borgen',  
      'Ash vs Evil Dead', 'Heartland', 'Unbelievable',  
      'The Promised Neverland', 'Derry Girls', 'Innocent',  
      'Trollhunters: Tales of Arcadia', 'Outlander',  
      'The Legend of Korra', 'The Dark Crystal: Age of Resistance',  
      'Vincenzo', 'Top Boy', 'Babylon Berlin', 'Stargate SG-1',  
      'Violet Evergarden', 'Narcos: Mexico', 'The Dragon Prince', 'Maid',  
      'Car Masters: Rust to Riches', 'Naruto', 'The Originals', 'Fauda',  
      'Sex Education', 'Kingdom', 'Money Heist', 'Young Royals',  
      'Atypical', 'Gurren Lagann', '30 Rock', 'Castlevania',  
      'Kim's Convenience', 'Master of None', 'Longmire',  
      'Call My Agent!', 'The Get Down', 'Sense8', 'One Day at a Time',  
      'The Kominsky Method', 'Itaewon Class', 'The Good Place',  
      'Grace and Frankie', 'The Witcher', 'Locked Up', 'Gilmore Girls',  
      'Caliphate', 'Fate/Zero', 'The Walking Dead',  
      'anohana: The Flower We Saw That Day',  
      'The Trials of Gabriel Fernandez', 'How to Get Away with Murder',  
      'Derek', 'Rascal Does Not Dream of Bunny Girl Senpai',  
      'Wild Wild Country', 'Bodyguard', 'American Vandal',
```



+ Code + Text

Connect | Editing

```
[ ] #10- NUMBER OF SHOWS INTO 4 DIFFERENT CATAGORY ACCORDING TO THEIR RATINGS/SCORE
import numpy as np
masterpiece = np.sum(df['score']>=9.0) #9.0 to 10 RATING
delicate = np.sum((df['score']>=8.0)&(df['score']<8.9)) #8.0 to 8.9 RATING
good = np.sum((df['score']>=7.0)&(df['score']<7.9)) #7.0 to 7.9 RATING
boring = np.sum((df['score']>=5.0)&(df['score']<6.9)) #5.0 to 6.9 RATING
print('Masterpiece',masterpiece,'\nDelicate',delicate,'\nGood',good,'\nBoring',boring)
```

```
Masterpiece 9
Delicate 120
Good 88
Boring 0
```

```
[ ] #11- GROUP BY THE DATA 'GENRE' AND CALCULATE THE MEAN OF EACH GROUP
df.groupby('main_genre').mean()
```

	release_year	score	number_of_votes	duration	number_of_seasons
main_genre					
action	2014.964286	8.085714	128604.642857	42.642857	4.500000
animation	2015.500000	8.175000	46908.000000	26.750000	3.000000
comedy	2012.697674	8.069767	76711.744186	32.465116	4.232558
crime	2017.300000	7.985000	91137.550000	52.050000	2.150000
documentary	2019.000000	8.357143	41390.285714	46.000000	2.000000
drama	2016.097561	8.113415	105265.207317	45.658537	3.463415
reality	2018.000000	8.450000	14085.500000	43.000000	4.500000
romance	2015.000000	7.700000	10102.000000	22.000000	4.000000
scifi	2012.266667	8.115556	139158.133333	38.577778	3.800000





+ Code + Text

Connect ▾

Editing



thriller	2015.750000	7.850000	111789.000000	49.750000	5.000000
war	2016.125000	7.875000	44704.875000	48.375000	3.250000
western	2005.000000	8.600000	73624.500000	39.000000	3.500000



```
[ ] #12-GROIUP BY DATA BY 'GENRE' AND COUNT THE NUMBER OF THE ROWS IN EACH GROUP  
df.groupby('main_genre').count()
```

	title	release_year	score	number_of_votes	duration	number_of_seasons	main_production
main_genre							
action	28	28	28	28	28	28	28
animation	4	4	4	4	4	4	4
comedy	43	43	43	43	43	43	43
crime	20	20	20	20	20	20	20
documentary	7	7	7	7	7	7	7
drama	82	82	82	82	82	82	82
reality	2	2	2	2	2	2	2
romance	1	1	1	1	1	1	1
scifi	45	45	45	45	45	45	45
thriller	4	4	4	4	4	4	4
war	8	8	8	8	8	8	8
western	2	2	2	2	2	2	2

co

MajorProject2.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment

Share

+ Code + Text

Connect Editing

[ ] #13- THE HIGHEST SCORE/RATING A SHOW HAS GOT  
np.max(df['score'])  
  
9.5

[ ] #14- THE LOWEST SCORE/RATING A SHOW HAS GOT  
np.min(df['score'])  
  
7.5

[ ] #15- SHOW THE SUMMERY STATISTICS OF THE SHOWS  
df.describe()

	release_year	score	number_of_votes	duration	number_of_seasons
count	246.000000	246.000000	2.460000e+02	246.000000	246.000000
mean	2014.760163	8.093496	1.019668e+05	41.918699	3.650407
std	6.392869	0.449261	1.688746e+05	15.242579	3.191785
min	1969.000000	7.500000	1.002400e+04	16.000000	1.000000
25%	2013.000000	7.700000	1.855325e+04	27.250000	1.000000
50%	2016.500000	8.000000	4.194250e+04	44.000000	3.000000
75%	2019.000000	8.400000	1.128118e+05	51.750000	5.000000
max	2022.000000	9.500000	1.727694e+06	141.000000	21.000000